Bits Practise Contest 4

A. Beautiful Array

2 seconds, 256 megabytes

You are given an array a consisting of n integers. Beauty of array is the maximum sum of some **consecutive subarray** of this array (this subarray may be empty). For example, the beauty of the array `[10, -5, 10, -4, 1]` is `15`, and the beauty of the array `[-3, -5, -1]` is `0`.

You may choose **at most one consecutive subarray** of a and multiply all values contained in this subarray by x. You want to maximize the beauty of array after applying at most one such operation.

Input

The first line contains two integers n and x ($1 \le n \le 3 \cdot 10^5, -100 \le x \le 100$) — the length of array a and the integer x respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the array a.

Output

Print one integer — the maximum possible beauty of array a after multiplying all values belonging to some consecutive subarray x.

input
5 -2 -3 8 -2 1 -6
output
22

input
12 -3 1 3 3 7 1 3 3 7 1 3 3 7
output
42

input
5 10 -1 -2 -3 -4 -5
output
0

In the first test case we need to multiply the subarray `[-2, 1, -6]`, and the array becomes `[-3, 8, 4, -2, 12]` with beauty `22` (`[-3, 8, 4, -2, 12]`).

In the second test case we don't need to multiply any subarray at all.

In the third test case no matter which subarray we multiply, the beauty of array will be equal to `0`.

B. Yet Another Array Queries Problem

2 seconds, 256 megabytes

You are given an array a of size n, and q queries to it. There are queries of two types:

- $1\ l_i\ r_i$ — perform a cyclic shift of the segment $[l_i, r_i]$ to the right. That is, for every x such that $l_i \le x < r_i$ new value of a_{x+1} becomes equal to old value of a_x, and new value of a_{l_i} becomes equal to old value of a_{r_i};
- $2\ l_i\ r_i$ — reverse the segment $[l_i, r_i]$.

There are m important indices in the array $b_1, b_2, \ldots, b_m$. For each i such that $1 \le i \le m$ you have to output the number that will have index b_i in the array after all queries are performed.

Input

The first line contains three integer numbers n, q and m ($1 \le n, q \le 2 \cdot 10^5$, $1 \le m \le 100$).

The second line contains n integer numbers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Then q lines follow. i-th of them contains three integer numbers t_i, l_i, r_i, where t_i is the type of i-th query, and $[l_i, r_i]$ is the segment where this query is performed ($1 \le t_i \le 2$, $1 \le l_i \le r_i \le n$).

The last line contains m integer numbers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le n$) — important indices of the array.

Output

Print m numbers, i-th of which is equal to the number at index b_i after all queries are done.

input
6 3 5 1 2 3 4 5 6 2 1 3 2 3 6 1 1 6 2 2 1 5 3
output
3 3 1 5 2

C. Minimal Segment Cover

2 seconds, 256 megabytes

You are given n intervals in form $[l; r]$ on a number line.

You are also given m queries in form $[x; y]$. What is the minimal number of intervals you have to take so that every point (**not necessarily integer**) from x to y is covered by at least one of them?

If you can't choose intervals so that every point from x to y is covered, then print `-1` for that query.

Input

The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of intervals and the number of queries, respectively.

Each of the next n lines contains two integer numbers l_i and r_i ($0 \le l_i < r_i \le 5 \cdot 10^5$) — the given intervals.

Each of the next m lines contains two integer numbers x_i and y_i ($0 \le x_i < y_i \le 5 \cdot 10^5$) — the queries.

Output

Print m integer numbers. The i-th number should be the answer to the i-th query: either the minimal number of intervals you have to take so that every point (**not necessarily integer**) from x_i to y_i is covered by at least one of them or `-1` if you can't choose intervals so that every point from x_i to y_i is covered.

input
2 3 1 3 2 4 1 3 1 4 3 4
output
1 2 1

input

```
3 4
1 3
1 3
4 5
1 2
1 3
1 4
1 5
```

output

```
1
1
-1
-1
```

In the first example there are three queries:

1. query $[1; 3]$ can be covered by interval $[1; 3]$;
2. query $[1; 4]$ can be covered by intervals $[1; 3]$ and $[2; 4]$. There is no way to cover $[1; 4]$ by a single interval;
3. query $[3; 4]$ can be covered by interval $[2; 4]$. It doesn't matter that the other points are covered besides the given query.

In the second example there are four queries:

1. query $[1; 2]$ can be covered by interval $[1; 3]$. Note that you can choose any of the two given intervals $[1; 3]$;
2. query $[1; 3]$ can be covered by interval $[1; 3]$;
3. query $[1; 4]$ can't be covered by any set of intervals;
4. query $[1; 5]$ can't be covered by any set of intervals. Note that intervals $[1; 3]$ and $[4; 5]$ together don't cover $[1; 5]$ because even non-integer points should be covered. Here 3.5, for example, isn't covered.

D. Little Elephant and Tree

4 seconds, 256 megabytes

The Little Elephant loves trees very much, he especially loves root trees.

He's got a tree consisting of n nodes (the nodes are numbered from 1 to n), with root at node number 1. Each node of the tree contains some list of numbers which initially is empty.

The Little Elephant wants to apply m operations. On the i-th operation $(1 \le i \le m)$ he first adds number i to lists of all nodes of a subtree with the root in node number a_i, and then he adds number i to lists of all nodes of the subtree with root in node b_i.

After applying all operations the Little Elephant wants to count for each node i number c_i — the number of integers j $(1 \le j \le n; j \ne i)$, such that the lists of the i-th and the j-th nodes contain at least one common number.

Help the Little Elephant, count numbers c_i for him.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^5)$ — the number of the tree nodes and the number of operations.

Each of the following $n - 1$ lines contains two space-separated integers, u_i and v_i $(1 \le u_i, v_i \le n, u_i \ne v_i)$, that mean that there is an edge between nodes number u_i and v_i.

Each of the following m lines contains two space-separated integers, a_i and b_i $(1 \le a_i, b_i \le n, a_i \ne b_i)$, that stand for the indexes of the nodes in the i-th operation.

It is guaranteed that the given graph is an undirected tree.

Output

In a single line print n space-separated integers — $c_1, c_2, ..., c_n$.

input

```
5 1
1 2
1 3
3 5
3 4
2 3
```

output

```
0 3 3 3 3
```

input

```
11 3
1 2
2 3
2 4
1 5
5 6
5 7
5 8
6 9
8 10
8 11
2 9
3 6
2 8
```

output

```
0 6 7 6 0 2 0 5 4 5 5
```

E. The Fair Nut and the Best Path

3 seconds, 256 megabytes

The Fair Nut is going to travel to the Tree Country, in which there are n cities. Most of the land of this country is covered by forest. Furthermore, the local road system forms a tree (connected graph without cycles). Nut wants to rent a car in the city u and go by a simple path to city v. He hasn't determined the path, so it's time to do it. Note that chosen path can consist of only one vertex.

A filling station is located in every city. Because of strange law, Nut can buy only w_i liters of gasoline in the i-th city. We can assume, that he has **infinite money**. Each road has a length, and as soon as Nut drives through this road, the amount of gasoline decreases by length. Of course, Nut can't choose a path, which consists of roads, where he runs out of gasoline. He can buy gasoline in **every** visited city, even in **the first** and **the last**.

He also wants to find the maximum amount of gasoline that he can have at the end of the path. Help him: count it.

Input

The first line contains a single integer n $(1 \le n \le 3 \cdot 10^5)$ — the number of cities.

The second line contains n integers $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 10^9$) — the maximum amounts of liters of gasoline that Nut can buy in cities.

Each of the next $n - 1$ lines describes road and contains three integers u, v, c $(1 \le u, v \le n, 1 \le c \le 10^9, u \ne v)$, where u and v — cities that are connected by this road and c — its length.

It is guaranteed that graph of road connectivity is a tree.

Output

Print one number — the maximum amount of gasoline that he can have at the end of the path.

input

```
3
1 3 3
1 2 2
1 3 2
```

output

```
3
```

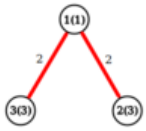
input

```
5
6 3 2 5 0
1 2 10
2 3 3
2 4 1
1 5 1
```

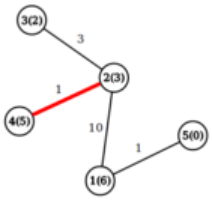
output

```
7
```

The optimal way in the first example is $2 \to 1 \to 3$.



The optimal way in the second example is $2 \to 4$.



F. Happy Tree Party

3 seconds, 256 megabytes

Bogdan has a birthday today and mom gave him a tree consisting of n vertecies. For every edge of the tree i, some number x_i was written on it. In case you forget, a tree is a connected non-directed graph without cycles. After the present was granted, m guests consecutively come to Bogdan's party. When the i-th guest comes, he performs exactly one of the two possible operations:

1. Chooses some number y_i, and two vertecies a_i and b_i. After that, he moves along the edges of the tree from vertex a_i to vertex b_i using the shortest path (of course, such a path is unique in the tree). Every time he moves along some edge j, he replaces his current number y_i by $y_i = \lfloor \frac{y_i}{x_j} \rfloor$, that is, by the result of integer division y_i div x_j.
2. Chooses some edge p_i and replaces the value written in it x_{p_i} by some positive integer $c_i < x_{p_i}$.

As Bogdan cares about his guests, he decided to ease the process. Write a program that performs all the operations requested by guests and outputs the resulting value y_i for each i of the first type.

Input

The first line of the input contains integers, n and m ($2 \le n \le 200\,000$, $1 \le m \le 200\,000$) — the number of vertecies in the tree granted to Bogdan by his mom and the number of guests that came to the party respectively.

Next $n - 1$ lines contain the description of the edges. The i-th of these lines contains three integers u_i, v_i and x_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$, $1 \le x_i \le 10^{18}$), denoting an edge that connects vertecies u_i and v_i, with the number x_i initially written on it.

The following m lines describe operations, requested by Bogdan's guests. Each description contains three or four integers and has one of the two possible forms:

- $1\ a_i\ b_i\ y_i$ corresponds to a guest, who chooses the operation of the first type.
- $2\ p_i\ c_i$ corresponds to a guests, who chooses the operation of the second type.

It is guaranteed that all the queries are correct, namely $1 \le a_i, b_i \le n$, $1 \le p_i \le n - 1$, $1 \le y_i \le 10^{18}$ and $1 \le c_i < x_{p_i}$, where x_{p_i} represents a number written on edge p_i at this particular moment of time that is not necessarily equal to the initial value x_{p_i}, as some decreases may have already been applied to it. The edges are numbered from 1 to $n - 1$ in the order they appear in the input.

Output

For each guest who chooses the operation of the first type, print the result of processing the value y_i through the path from a_i to b_i.

input

```
6 6
1 2 1
1 3 7
1 4 4
2 5 5
2 6 2
1 4 6 17
2 3 2
1 4 6 17
1 5 5 20
2 4 1
1 5 1 3
```

output

```
2
4
20
3
```

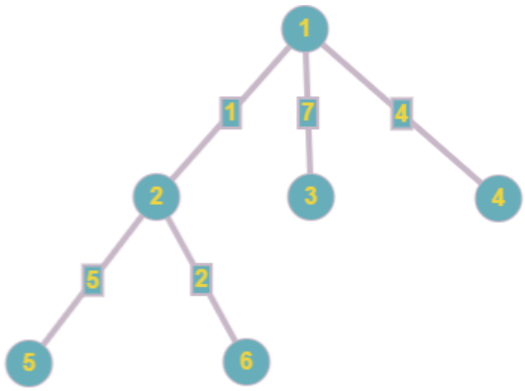
input

```
5 4
1 2 7
1 3 3
3 4 2
3 5 5
1 4 2 100
1 5 4 1
2 2 2
1 1 3 4
```

output

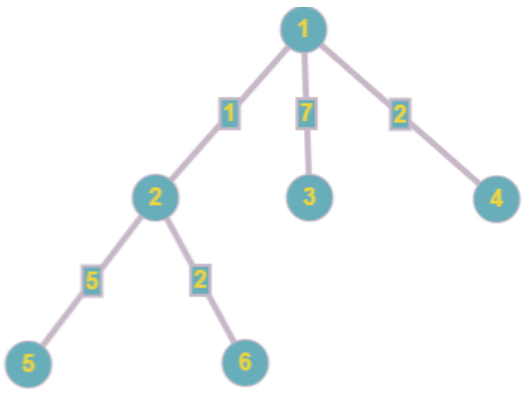
```
2
0
2
```

Initially the tree looks like this:



The response to the first query is: $\lfloor \frac{\lfloor \frac{17}{4} \rfloor}{2} \rfloor = 2$

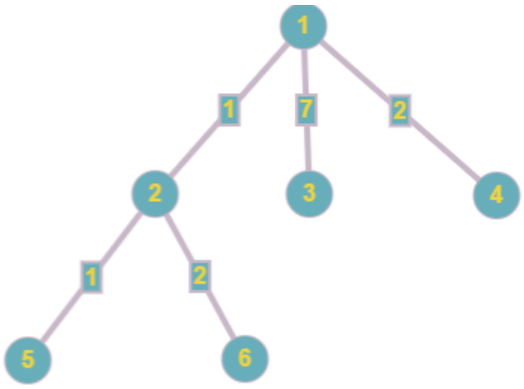
After the third edge is changed, the tree looks like this:



The response to the second query is: $\left\lfloor \frac{\lfloor \frac{11}{2} \rfloor}{2} \right\rfloor = 4$

In the third query the initial and final vertex coincide, that is, the answer will be the initial number 20.

After the change in the fourth edge the tree looks like this:



In the last query the answer will be: $\left\lfloor \frac{\lfloor \frac{3}{1} \rfloor}{1} \right\rfloor = 3$

G. New Year Tree

2 seconds, 256 megabytes

You are a programmer and you have a New Year Tree (not the traditional fur tree, though) — a tree of four vertices: one vertex of degree three (has number 1), connected with three leaves (their numbers are from 2 to 4).

On the New Year, programmers usually have fun. You decided to have fun as well by adding vertices to the tree. One adding operation looks as follows:

- First we choose some leaf of the tree with number v.
- Let's mark the number of vertices on the tree at this moment by variable n, then two vertexes are added to the tree, their numbers are $n + 1$ and $n + 2$, also you get new edges, one between vertices v and $n + 1$ and one between vertices v and $n + 2$.

Your task is not just to model the process of adding vertices to the tree, but after each adding operation print the diameter of the current tree. Come on, let's solve the New Year problem!

Input

The first line contains integer q $(1 \le q \le 5 \cdot 10^5)$ — the number of operations. Each of the next q lines contains integer v_i $(1 \le v_i \le n)$ — the operation of adding leaves to vertex v_i. Variable n represents the number of vertices in the current tree.

It is guaranteed that all given operations are correct.

Output

Print q integers — the diameter of the current tree after each operation.

input
5 2 3 4 8 5

output
3 4 4 5 6

H. Appleman and a Sheet of Paper

2 seconds, 256 megabytes

Appleman has a very big sheet of paper. This sheet has a form of rectangle with dimensions $1 \times n$. Your task is help Appleman with folding of such a sheet. Actually, you need to perform q queries. Each query will have one of the following types:

1. Fold the sheet of paper at position p_i. After this query the leftmost part of the paper with dimensions $1 \times p_i$ must be above the rightmost part of the paper with dimensions $1 \times ([\textit{current width of sheet}] - p_i)$.
2. Count what is the total width of the paper pieces, if we will make two described later cuts and consider only the pieces between the cuts. We will make one cut at distance l_i from the left border of the current sheet of paper and the other at distance r_i from the left border of the current sheet of paper.

Please look at the explanation of the first test example for better understanding of the problem.

Input

The first line contains two integers: n and q $(1 \le n \le 10^5; 1 \le q \le 10^5)$ — the width of the paper and the number of queries.

Each of the following q lines contains one of the described queries in the following format:

- "$1\ p_i$" $(1 \le p_i < [\textit{current width of sheet}])$ — the first type query.
- "$2\ l_i\ r_i$" $(0 \le l_i < r_i \le [\textit{current width of sheet}])$ — the second type query.

Output

For each query of the second type, output the answer.

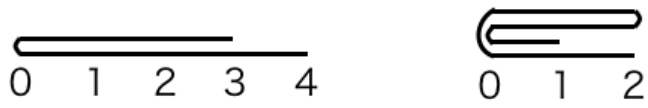
input
7 4 1 3 1 2 2 0 1 2 1 2

output
4 3

input
10 9 2 2 9 1 1 2 0 1 1 8 2 0 8 1 2 2 1 3 1 4 2 2 4

output
7 2 10 4 5

The pictures below show the shapes of the paper during the queries of the first example:



After the first fold operation the sheet has width equal to 4, after the second one the width of the sheet equals to 2.